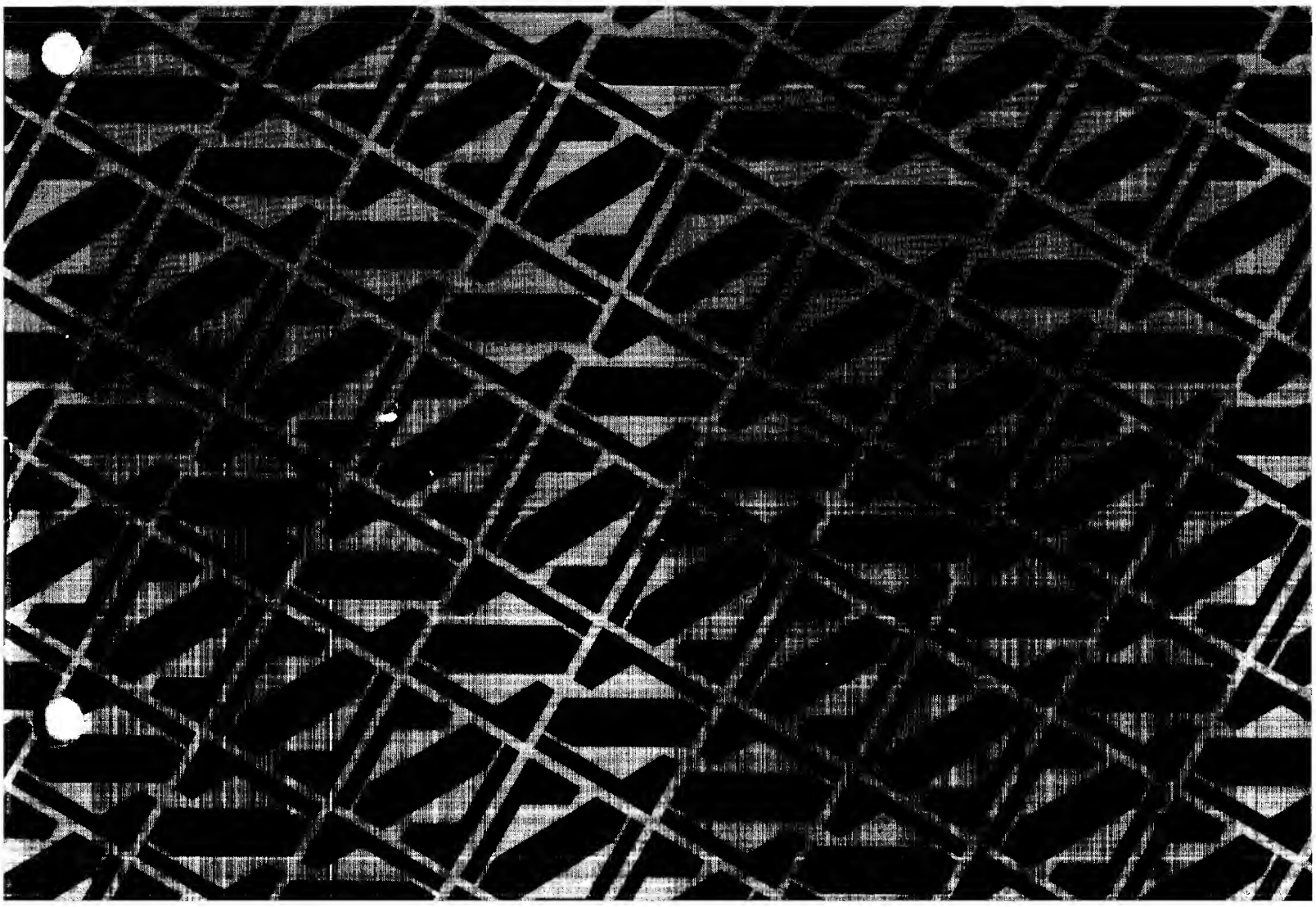
National Semiconductor
Order No. IMP-16S/048YA
Pub. No. 4200048A

IMP-16C DEBUGC Utility Program Reference Manual



Order Number IMP-16S/048YA
Publication Number 4200048A

Integrated MicroProcessor-16C

IMP-16C

DEBUGC UTILITY PROGRAM

REFERENCE MANUAL

March 1974

2900 Semiconductor Drive
Santa Clara, California 95051

PREFACE

This publication provides information pertaining to the IMP-16C DEBUGC Utility Program. The DEBUGC language, communications requirements and procedures, and control statements are described. The DEBUGC listing is presented as appendix A.

The material in this manual is for information purposes only and is subject to change without notice.

Copies of this publication and other National Semiconductor publications may be obtained from the sales offices listed on the back cover.

CONTENTS

ILLUSTRATIONS

TABLES

DEBUGC UTILITY PROGRAM

1.0 INTRODUCTION

DEBUGC is a firmware program that supervises the operation of a user's program during checkout. The user has the facility to enter a set of commands to the DEBUGC program, through a Teletype keyboard, to which the DEBUGC program responds by performing the requested action and communicating data back to the user through the Teletype printer. DEBUGC provides the following facilities for testing and running the user's programs in the IMP-16C.

- Printing selected areas of memory in hexadecimal format.
- Modifying the contents of selected areas in memory.
- Modifying processor registers and the top word of the stack.
- Inserting instruction breakpoint halts.
- Initiating execution at any point in program.

The DEBUGC listing is given in appendix A.

2.0 CONFIGURATION AND USAGE

The minimum system configuration needed is an IMP-16C, a control panel, and a Teletype. A simple Teletype interface circuit as described in the IMP-16 Application Manual, Supplement 1 (or IMP-16 Interfacing Guide), is also needed. DEBUGC is supplied as a set of two ROMs designed to be used with two CUTIL ROMs (IMP-16C Utilities program). In this mode of use, the CUTIL ROMs are inserted in the memory range $FF00_{16}$ to $FFFF_{16}$, and DEBUGC occupies the range $FE00_{16}$ through $FEFF_{16}$. DEBUGC uses location 0 and locations 2 through X'E of base page; these locations must not be altered by the user while DEBUGC is being used.

2.1 DEBUGC LANGUAGE

The control statements which are used to command the operation of DEBUGC are confined within a set of rules which define the syntax (the format of control statements), and semantics (the meanings of the various symbols and characters comprising the control statement) of the language.

2.1.2 Conventions Used in This Manual

The following notation is used, both in the general cases (in the command descriptions) and in the specific cases (in the examples):

- Mixed upper- and lower-case characters are used for comments and notes.
- Nonunderlined characters, numbers, and symbols, used in the examples, indicate computer-generated output from the Teletype printer. For example, memory contents appear as follows:

```
0100    7890    2413    0016
```

- Underlined characters, numbers, and symbols, used in the examples indicate user-generated input at the Teletype keyboard. Two classes of statements are underlined, lower-case and upper-case as follows:

  Lower-case statements or statement parts represent the general case (to be further defined by the rules of syntax).

  Upper-case statements or statement parts represent the exact (specific) form of the input required to be typed in.

  For example:

> T <address argument>	(general case)
> T 2345:2375	(specific case)
> NOTE ADDRESS	(specific case)

  Circled upper-case characters represent operation of Teletype keyboard keys that do not generate a printed character.

  For example: (CR) represents the carriage return key.
  (LF) represents the line feed key.

2.1.3 Syntax and Semantics

The basic elements of DEBUGC commands are defined below. In the formal (symbolic) descriptions of DEBUGC commands, the following symbols are used:

<a>	Specifies an element 'a' either of a command or of another element.
: :=	Means 'is defined as' and appears in a statement which defines the element to its left.
{a, b, c}	Indicates that one of the elements specified inside the braces must be included in the statement.
[a]	Indicates that the element(s) specified within the brackets are optional and need not be included in the command, unless desired.

2.1.4 Syntax

The following meanings are assigned to the terms used in the general-case form of the statements:

<hexadecimal number> : :=	From one to four digits from the hexadecimal set (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). Leading zeros may be omitted. If more than four digits are entered, only the last four are used.
<value> : :=	A four-digit hexadecimal number used as the contents of a memory location or the contents of a register. Consists of a 16-bit number.
<memory address> : :=	A four-digit hexadecimal number specifying a memory location. Leading zeros may be omitted.
<memory address range> : :=	A memory address, followed by a colon (:), followed by a second memory address. For example: 3528:354A

The memory address to the left of the colon represents the low limit of the range; the address to the right of the colon represents the high limit of the range. If the upper limit of the range is smaller than the lower limit of the range, DEBUGC accepts only the lower number and performs the requested operation at that address.

<register address> — A register address may be selected from the set of the following:
0, 1, 2, 3, 4, 5

0 represents AC0
1 represents AC1
2 represents AC2
3 represents AC3
4 represents Flag Register
5 represents Top Word of Stack

<comment> — English language text, including letters and numbers, exactly as typed in.

2.1.5 Semantics

All numbers input to DEBUGC may be either decoded as hexadecimal or used in the NOTE command in ASCII format. The following description explains the use of certain characters:

: (colon) — Delimiter for a range argument. Signifies that all the locations from the first entry through the last are included in the range; that is, a:b signifies all the locations from a through b, including a and b.

, (comma) — Delimiter of address and range arguments.

3.0 COMMUNICATIONS

The user can communicate with DEBUGC through a Teletype. Whenever DEBUGC takes control, it types the 'greater than' character (>) to indicate that it is ready to accept a command. The user then may type control statements to direct the operation of DEBUGC. All commands must be terminated by a carriage return (CR) or a line feed (LF). To abort a command, the (ALT MODE) key may be pressed at any time before the (LF) or (CR); the (?) symbol is printed and no further action occurs. Blanks have no significance and may be inserted anywhere; the null and rubout characters also are ignored. If DEBUGC detects an error in a command, it types a question mark (?) and prompts for a new command by typing the prompt character (>).

Control is returned to DEBUGC from a user's program by use of the HALT command. DEBUGC types the prompt (>) upon reentry. Control is transferred back to the user's program from DEBUGC by the GO (G) directive. Details pertaining to the HALT and GO directives are described under the descriptions of the commands.

4.0 CONTROL STATEMENTS

A control statement consists of a single alphabetic character representing the command to be performed, followed by a parameter list of the arguments for the commanded operation; the arguments are separated by commas. The numeric fields in a parameter list must be in hexadecimal notation; leading zeros may be

omitted. Overflow is not checked; only the last four digits entered are used. This feature may be used to correct typing errors without retyping the whole line. Blank characters are ignored, as are the null and rubout characters. A statement must be terminated with a carriage return (CR) or a line feed (LF)

In the examples that follow, information which is input by the user is shown underlined. A summary of the commands and the statement format is listed in table 4-1.

Table 4-1. Summary of DEBUGC Commands

Command	Statement Format
ALTER	A <memory address> , <value> [, <value>...]
	or A, <value> [, <value> ...]
GO	G <memory address>
HALT	H <memory address>
INSERT	I <value> , <memory address range>
LOAD REGISTER	L <register address> , <value> [, <value>...]
REGISTER TYPE	R
TYPE	T { <memory address> / <memory address range> }

4.1 TYPE

T { <memory address> / <memory address range> }

The contents of the specified locations are printed on the terminal in hexadecimal notation. For each line of the printout, the starting address is printed, followed by one to eight locations per line. The address for a new line is always a multiple of eight for consistency and ease of reading. The first line may contain fewer than eight locations if the starting address is not a multiple of eight. The final location referenced becomes the value of the current location (CL).

As information is transmitted to the Teletype, the Teletype is interrogated for input. If any character is detected, the output is terminated and the user is prompted for another DEBUGC command. This feature may be used for terminating an excessive or undesirable output.

The following example illustrates the use of the TYPE command. The first line following the prompt character (>) is input by the user; the following lines and the final prompt character are generated by DEBUGC and are output by the program:

```
>T100:10A (CR)
  0100   7890   1010   0001   0004   0005   0006   5555   0000
  0108   60FF   ABCD   1234

>
```

4.2 REGISTER TYPE

R

The contents of the four registers plus the flags and the top-of-stack are printed on the terminal in hexadecimal notation. The order to printout is as follows: AC0, AC1, AC2, AC3, FLAGS, TOP-OF-STACK. The following example illustrates the use of the REGISTER TYPE command. The user enters the character 'R', followed by a (CR) . DEBUGC generates the second line and the following prompt character:

```
>R  (CR)

  0000  FACE  CAFE  0ACE  BADE  FADE

>
```

4.3 ALTER

A<memory address> , <value> [, <value> . . .] or
A , <value> [, <value> . . .]

The ALTER command alters the contents of memory beginning at the address specified. Each subsequent value is stored in the next higher location. A null field (two commas with no intervening number) leaves the corresponding memory location unaltered. If the memory address (first) field is null (no number), alteration commences with the current location. The current location is the location following the last location altered. The following example illustrates use of the ALTER command. The TYPE commands are included to show the data change in the specified memory ranges. The characters on each line following the prompt character are input by the user; the tabulated data is generated by DEBUGC.

```
>A 100,1,2,3  (CR)
>A  ,4,5,6  (CR)
>T 100:107  (CR)

  0100  0001  0002  0003  0004  0005  0006  0000  0000

>A 100,1111,2222,,,,5555  (CR)
>T 100:107  (CR)

  0100  1111  2222  0003  0004  0005  5555  0000  0000

>
```

When the ALTER command is terminated with use of the (ALT MODE) key, the operation is aborted at the current value being entered. Typing the comma, following the entry of a value, causes the value to be stored in the specified location. If the last value before the (ALT MODE) is followed with a comma, it is used; if the value is not followed with a comma, the ALTER command is aborted before the value is used to alter the specified location. The (ALT MODE) does not abort the entire command.

For example:

```
>A 100,1,2,3,7 (ALT) ?
>T 100:103 (CR)
  0100   0001   0002   0003   3F08*
>
```

*Unaltered prior value.

4.4 LOAD REGISTER

L <register address>, <value> [, <value>...]

The LOAD REGISTER command works in exactly the same manner as the ALTER command, except that it is the value stored in a register that is changed rather than a value in main memory. Care must be taken to ensure that the parameter list of values does not extend beyond the register save area. The allowed register addresses are as follows:

0	AC0
1	AC1
2	AC2
3	AC3
4	STATUS FLAGS
5	TOP-OF-STACK

The following example is a sample of user input, specifying the LOAD REGISTER command:

```
>L 0,0,1,2,3,4 (CR)
>
```

The register values are stored in memory until the GO instruction is executed. The LOAD REGISTER command alters these saved values. The SELECT Flag may be examined in memory location 0008, using the TYPE command, and may be altered via the ALTER command. A value of 0 or 1 must be used.

4.5 INSERT

I <value>, <memory address range>

The INSERT command may be used to insert a value in a selected memory location. The original word, in the selected location, and all subsequent words within the defined range are moved up one word; the new data word is then inserted in the first location of the range, which is the selected location. Care must be exercised when using this command to insert a word in a block of instructions because forward and backward address references will be changed within the particular program segment.

The range specification must contain two addresses. The data from the first location to the last location (inclusive) is moved up one; that is, $(x + 1) \leftarrow (x)$. The first address specified is then replaced with the hexadecimal number (value). The data in the location following the specified range will be lost.

The following example illustrates use of the INSERT command. The user enters the command following the prompt character. The TYPE command is included to display the contents of the specified range before and after the INSERT command is used.

```
>T 100:107 (CR)
  0100   7890   0001   0003   0004   0005   5555   3300   FF00
>I 1010, 101:105 (CR)
>T 100:107 (CR)
  0100   7890   1010   0001   0003   0004   0005   5555   FF00
>
```

4.6 HALT

H <memory address>

The HALT command terminates control by the user's program at the location specified and returns control to DEBUGC. HALT causes the program to terminate just before the memory address specified in the command. The instruction is subsequently executed immediately after control is returned to the user's program by use of the GO command. The halt location must be in read/write memory; no halt occurs during execution if the location specified is in ROM.

The HALT command works by exchanging the instruction at the given location for a JMP (jump) to DEBUGC, saving the original instruction for later execution after the GO command. Only one HALT may by in use at a time; a subsequent HALT command resets the original location and sets a new breakpoint halt location. The HALT may be removed and no new HALT set by a HALT command with a zero address (H0).

A GO command following the HALT command is successfully executed only when the instruction at the HALT location and the instruction at the following location always are executed consecutively; or when the instruction is a base page or indexed jump (but not a PC-relative jump). Other PC-relative instructions do work, but the operand is always wrong (since the instruction is not actually executed in its original location).

4.7 GO

G <memory address>

The GO command starts execution of the user's program at the location specified by the memory address operand of the GO statement or (by default when this operand field is not used) at the memory address of the last executed HALT command. However, if the last halt instruction was an H0 command, removing

the active halt command, an error condition occurs when the GO command attempts to use this address (see HALT command, paragraph 4.6). DEBUGC flags this condition and returns to the command mode by typing a new prompt character (>). If when first turned on, prior to using any HALT commands, a GO command without a memory address operand is used, unpredictable results occur.

To avoid the situation where a jump to an unpredictable address may occur, it is suggested that the user first type in an H0 command to set the halt address to 0000. Thus an inadvertent GO command without an address will not cause the user's program to be destroyed in memory.

The four registers, the flags and the Select Flag all are restored when the GO command is given. The top location of the stack is saved and is also restored. The remaining locations in the stack are unaltered, but since DEBUGC does not save these locations of the stack, stack overflow may occur if the stack is more than half-full. Up to eight words in the bottom of the stack may be cleared during execution of DEBUGC.

The HALT and GO instructions may be used to step through a program one instruction (or a few) at a time.

For example, suppose the main program is as follows:

```
36 0040 4C00            LI      0,0
37 0041 2444            JSR     SUB
38 0042 0F80            PFLG    7
39 0043 0000            HALT
40 0044 xxxx    SUBR:   xxxx    xxxx
xx xxxx xxxx            xxxx    xxxx
xx xxxx xxxx            xxxx    xxxx
        0020            RTS
```

Single Stepping:

```
>R (CR)
  0102  0002  0FFF  0000  0000  5651
>H 0041 (CR)
>G  0040 (CR)
>R (CR)
  0000  0002  0FFF  0000  0000  5651
>N THE PREVIOUS COMMANDS EXECUTED THE (CR)
>N SINGLE INSTRUCTION AT 0040 (LI 0,0) (CR)
>H 42 (CR)
>G 41 (CR)
>N THE SUBROUTINE WAS JUST EXECUTED IN FULL. (CR)
>
```

4.8 NOTE

N <comment>

The NOTE command permits the user to comment his debugging. All text, up to the carriage return or line feed, is printed on the terminal. No other action is performed. The following are some examples of NOTE COMMENTS:

N *** INSERT INSTRUCTION *** (CR)

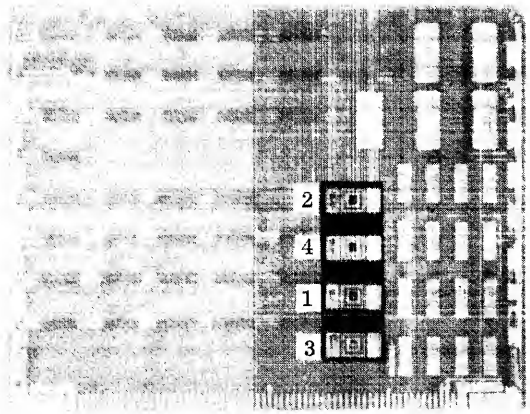
N ALTMODE CAUSES ABORT (CR)

NOTE THAT THE TTY PUNCH IS OFF (CR)

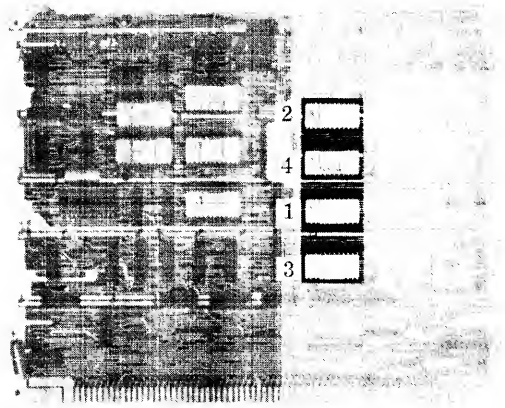
5.0 ROM LOCATIONS

Depending on which IMP-16 microprocessor is being used, the location of the ROMs containing DEBUGC and CUTIL are shown in figure 5-1.

IMP-16C/100



IMP-16C/200 and IMP-16C/300



Name	Address	Designator
1- CUTIL	FF00	IMP-16F/000BEW
2- CUTIL		IMP-16F/000BEX
3- DEBUG	FE00	IMP-16F/002BNR
4- DEBUG		IMP-16F/002BNS

Figure 5-1. DEBUG and CUTIL Device Locations

Appendix A

DEBUGC LISTING

```
 1 0000                 .TITLE  DEBUGC,'00310B   02/20/74'
 2 0000          ;*****************************************************************
 3 0000          ;*                                                               *
 4 0000          ;*      DEBUG  FOR THE IMP-16C  -  256 INSTRUCTION VERSION       *
 5 0000          ;*                                                               *
 6 0000          ;*      STARTING ADDRESS = OFE00  -  USE WITH  CUTIL  IN OFF00   *
 7 0000          ;*                                                               *
 8 0000          ;*****************************************************************
 9 0000                 .ASECT
10 0000 FE00 A  .       =       OFE00
11 FE00         ;
12 FE00         ;       DEFINITIONS
13 FE00         ;
14 FE00 0000 A  AC0     =       0
15 FE00 0001 A  AC1     =       1
16 FE00 0002 A  AC2     =       2
17 FE00 0003 A  AC3     =       3
18 FE00         ;
19 FE00 0001 A  ZRO     =       1               ; AC0 = 0
20 FE00 0002 A  POS     =       2               ; AC0 >= 0
21 FE00 0005 A  NZRO    =       5               ; AC0 ¬= 0
22 FE00 000B A  NEG     =       11              ; AC0 <= 0
23 FE00 000E A  JC14    =       14              ; TELETYPE INPJT JUMP CONDITION
24 FE00         ;
25 FE00         ;
26 FE00         ;       BASE PAGE ADDRESSES
27 FE00         ;
28 FE00 0002 A  SAV0    =       2               ; REGISTER STORAGE LOCATIONS
29 FE00 0003 A  SAV1    =       3
30 FE00 0004 A  SAV2    =       4
31 FE00 0005 A  SAV3    =       5
32 FE00 0006 A  FLAGS   =       6               ; SAVE FLAGS
33 FE00 0007 A  STACK   =       7               ; CONTENTS OF TOP OF STACK
34 FE00 0008 A  SELECT  =       8               ; SAVE SELECT FLAG
35 FE00 0009 A  POINTER =       9               ; POINTER TO BEGIN
36 FE00 000A A  CWRD    =       10              ; CURRENT WORD
37 FE00 000B A  HLOC    =       11              ; BREAKPOINT LOCATION
38 FE00 000C A  HDATA   =       12              ; BREAKPOINTED INSTRUCTION
39 FE00 000D A  RETLOC  =       13              ; INSERTION DATA OR JMP @RETADD
40 FE00 000E A  RETADD  =       14              ; RETURN ADDRESS
41 FE00 000D A  DATA    =       RETLOC


42 FE00                 .PAGE   'IMP16C DEBUGGING ROUTINE'
43 FE00         ;
44 FE00 A002 A  BEGIN:  ST      AC0,SAV0        ; SAVE REGISTERS IN
45 FE01 A403 A          ST      AC1,SAV1        ; LOCATIONS 2-5
46 FE02 A804 A          ST      AC2,SAV2
47 FE03 AC05 A          ST      AC3,SAV3
48 FE04 0080 A          PUSHF                   ; SAVE FLAGS
49 FE05 4400 A          PULL    AC0
50 FE06 A006 A          ST      AC0,FLAGS
51 FE07 4C01 A          LI      AC0,1           ; SAVE SELECT FLAG:
52 FE08 58EF A          ROR     AC0,17          ; 08000 IF OFF, 1 IF ON
53 FE09 6179 A          AND     AC0,D7          ; MASK LOWER BITS
54 FE0A AC08 A          ST      AC0,SELECT      ; 0 IF OFF, 1 IF ON
55 FE0B 5400 A          XCHRS   AC0             ; SAVE CONTENTS OF TOP OF STACK
```

```
56 FE0C A007 A          ST      AC0,STACK
57 FE0D                 ;
58 FE0D 2900 A PRCMPT:  JSR     .+1             ; GET ADDRESS OF BEGIN
59 FE0E 4400 A GETADD:  PULL    AC0             ; (SELF-RELOCATING METHOD)
60 FE0F 48F2 A          AISZ    AC0,BEGIN-GETADD
61 FE10 A009 A          ST      AC0,POINTER
62 FE11 815B A          LD      AC0,JUMPI
63 FE12 A000 A          ST      AC0,0           ; LOAD LOC 0 WITH JUMP TO DEBUG
64 FE13                 ;
65 FE13 2914 A          JSR     CRLF            ; PROMPT FOR COMMAND
66 FE14 4C3E A          LI      AC0,'>'/256
67 FE15 2D51 A          JSR     @SEND
68 FE16 2940 A          JSR     GECHO           ; FETCH COMMAND
69 FE17 21F5 A          JMP     PROMPT          ; IGNORE CR AND LF
70 FE18 4F2C A          LI      AC3,CTBL-BEGIN  ; TEST FOR COMMAND TYPE
71 FE19 CC09 A          ADD     AC3,POINTER     ; LOAD COMMAND TABLE ADDRESS
72 FE1A F300 A NXCOM:   SKNE    AC0,(AC3)       ; COMPARE WITH RECEIVED COMMAND
73 FE1B 2105 A          JMP     GOTOIT
74 FE1C 8700 A          LD      AC1,(AC3)
75 FE1D F51E A          SKNE    AC1,ZERO        ; CHECK FOR END OF TABLE
76 FE1E 2105 A          JMP     ERROR
77 FE1F 4B02 A          AISZ    AC3,2           ; CHECK NEXT COMMAND IF NOT EQUAL YET
78 FE20 21F9 A          JMP     NXCOM           ; TRY AGAIN
79 FE21 8F01 A GOTOIT:  LD      AC3,1(AC3)      ; CALCULATE ADDRESS OF PROPER ROUTINE
80 FE22 CC09 A          ADD     AC3,POINTER
81 FE23 2300 A          JMP     (AC3)           ; GO THERE
82 FE24                 ;
83 FE24 4C3F A ERROR:   LI      AC0,'?'/256     ; ERROR ROUTINE PRINTS A
84 FE25 2D41 A          JSR     @SEND           ; QUESTION MARK AND PROMPTS
85 FE26 2901 A          JSR     CRLF            ; FOR A NEW COMMAND
86 FE27 21E5 A          JMP     PROMPT
87 FE28                 ;
88 FE28 4C0D A CRLF:    LI      AC0,0D          ; SUBROUTINE TO SEND CR & LF
89 FE29 2944 A          JSR     SENDC
90 FE2A 4C0A A          LI      AC0,0A
91 FE2B 2142 A          JMP     SENDC           ; SEND CHARACTER AND RETURN
92 FE2C                 ;
93 FE2C                 ;
94 FE2C 0041 A CTBL:    .WORD   'A'/256,ALTER-BEGIN
   FE2D 003D A
95 FE2E 004C A          .WORD   'L'/256,LDREG-BEGIN
   FE2F 004F A
96 FE30 0054 A          .WORD   'T'/256,TYPE-BEGIN
   FE31 00C1 A
97 FE32 0052 A          .WORD   'R'/256,REGTYP-BEGIN
   FE33 00D3 A
98 FE34 0049 A          .WORD   'I'/256,INSERT-BEGIN
   FE35 00AF A
99 FE36 0048 A          .WORD   'H'/256,HALT-BEGIN
   FE37 00D9 A
100 FE38 0047 A         .WORD   'G'/256,GO-BEGIN
   FE39 00E6 A
101 FE3A 004E A         .WORD   'N'/256,NOTE-BEGIN
   FE3B 00BE A
102 FE3C 0000 A ZERO:   .WORD   0               ; END OF TABLE
```

```
                        .PAGE   'COMMAND PROCESSING'
                ;*******************************************************************
                ;*                                                                 *
                ;*      ALTER MEMORY LOCATIONS                                     *
                ;*                                                                 *
                ;*      A XX,YY,YY,YY,....      STORE DATA YY BEGINNING AT XX      *
                ;*      A ,YY,YY,YY,....        STORE DATA YY BEGINNING AT CURR ADDR *
                ;*                                                                 *
                ;*******************************************************************
                ;
2919 A  ALTER:  JSR     GECHO           ; CHECK FOR INITIAL COMMA
21E5 A          JMP     ERROR
F15A A          SKNE    ACO,COMMA
2104 A          JMP     ALTER3          ; USE CURRENT CWRD
                ;
4D00 A  ALTER1: LI      AC1,0           ; GET MEMORY ADDRESS
2944 A          JSR     GETHXA
11E0 A          BOC     ZRO,ERROR
A40A A  ALTER2: ST      AC1,CWRD        ; STORE ADDRESS IN CWRD
2911 A  ALTER3: JSR     GECHO           ; GET NEXT CHARACTER
21C6 A          JMP     PROMPT
F152 A  CKCOM:  SKNE    ACO,COMMA       ; CHECK FOR CONSECUTIVE COMMAS
2103 A          JMP     CPROC
4D00 A          LI      AC1,0
293C A          JSR     GETHXA          ; GET DATA
840A A          ST      AC1,@CWRD       ; STORE DATA
780A A  CPROC:  ISZ     CWRD            ; INCREMENT MEMORY ADDRESS
15F7 A          BOC     NZRO,ALTER3     ; CONTINUE IF TERMINATOR WAS A COMMA
21BE A  EXIT:   JMP     PROMPT          ; EXIT UPON CR/LF
                ;
                ;
                ;*******************************************************************
                ;*                                                                 *
                ;*      LOAD REGISTER R WITH DATA XX                               *
                ;*                                                                 *
                ;*      L R,XX (,XX,XX,XX)      UP TO FIVE DATA MAY BE GIVEN       *
                ;*                                                                 *
                ;*******************************************************************
                ;
2934 A  LDREG:  JSR     GETHX           ; GET REGISTER NUMBER
11D3 A          BCC     ZRO,ERROR
4902 A          AISZ    AC1,2           ; LOC 2-5 ARE SAVED REGISTERS
3481 A          RCPY    AC1,ACO
1BD0 A          BOC     NEG,ERROR       ; CHECK FOR PROPER RANGE
E52E A          SKG     AC1,D7          ; (FLAGS MAY ALSO BE ALTERED)
21EE A          JMP     ALTER2          ; GO GET DATA TO STORE
21CD A          JMP     ERROR


                        .PAGE   'SUBROUTINES'
                        .LOCAL
                ;*******************************************************************
                ;*                                                                 *
                ;*      SUBROUTINE TO READ, ECHO AND CHECK CHARACTERS FOR CR/LF     *
                ;*                                                                 *
                ;*      RETURN:  RTS 0 IF CR/LF;  RTS 1 OTHERWISE                  *
                ;*                                                                 *
                ;*******************************************************************
                ;
2D0E A  GECHO:  JSR     @RECV           ; GET AND ECHO THE CHARACTER
610F A          AND     ACO,H7F
11FD A          BCC     ZRO,GECHO       ; IGNORE NULL
F10D A          SKNE    ACO,H7F
21FB A          JMP     GECHO           ; IGNORE RUBOUT
F10E A          SKNE    ACO,GALT
213A A          JMP     ABORT           ; ALTMODE: ABORT COMMAND
F10A A          SKNE    ACO,GCR
```

```
168 FE5F 21C8 A          JMP     CRLF            ; CR: SEND CRLF AND RTS 0
169 FE60 F109 A          SKNE    AC0,GLF
170 FE61 21C6 A          JMP     CRLF            ; LF: SEND CRLF AND RTS 0
171 FE62 290B A          JSR     SENDC           ; ECHO THE CHARACTER
172 FE63 F1C8 A          SKNE    AC0,BLNK
173 FE64 21F2 A          JMP     GECHO           ; IGNORE BLANK
174 FE65 0201 A          RTS     1               ; RETURN WITH CHARACTER IN AC0
175 FE66          ;
176 FE66 FF3D A RECV:    .WORD   0FF3D           ; RECV ADDRESS (IN CUTIL)
177 FE67 FF53 A SEND:    .WORD   0FF53           ; SEND ADDRESS (IN CUTIL)
178 FE68 C07F A H7F:     .WORD   07F             ; 7-BIT MASK, RUBOUT
179 FE69 000D A GCR:     .WORD   0D              ; CARRIAGE RETURN
180 FE6A 000A A GLF:     .WORD   0A              ; LINE FEED
181 FE6B C07D A GALT:    .WORD   07D             ; ALTMODE
182 FE6C 0020 A BLNK:    .WORD   ' '/256
183 FE6D 2409 A JUMPI:   JMP     @POINTER        ; JUMP INSTRUCTION TO BEGIN
184 FE6E          ;
185 FE6E 4300 A SENDC:   PUSH    AC3             ; SAVE REGISTERS AND SEND CHARACTER
186 FE6F 4000 A          PSH     AC0
187 FE70 2DF6 A          JSR     @SEND
188 FE71 4400 A          PULL    AC0
189 FE72 4700 A          PULL    AC3
190 FE73 0200 A          RTS     0


191 FE74                 .PAGE
192 FE74          ;*************************************************************************
193 FE74          ;*                                                                       *
194 FE74          ;*       PUTW - SEND HEX WORD IN AC2 TO TTY                              *
195 FE74          ;*                                                                       *
196 FE74          ;*       ONLY REGISTER AC3 IS UNDISTURBED                                *
197 FE74          ;*                                                                       *
198 FE74          ;*************************************************************************
199 FE74          ;
200 FE74 4C20 A PUTW:    LI      AC0,' '/256     ; SENDS HEX WORD IN AC2 TO TTY
201 FE75 29F8 A          JSR     SENDC           ; SEND BLANK FIRST
202 FE76 4D04 A          LI      AC1,4
203 FE77 3881 A P1:      RCPY    AC2,AC0
204 FE78 5E04 A          SHL     AC2,4
205 FE79 5CF4 A          SHR     AC0,12          ; 4 BITS NOW IN AC0
206 FE7A 4830 A          AISZ    AC0,'0'/256     ; CONVERT TO ASCII
207 FE7B E106 A          SKG     AC0,G9
208 FE7C 2101 A          JMP     .+2
209 FE7D 4807 A          AISZ    AC0,'A'-'9'/256-1
210 FE7E 29EF A          JSR     SENDC           ; SEND 1 CHARACTER
211 FE7F 49FF A          AISZ    AC1,-1          ; 4 CHARACTERS TOTAL
212 FE80 21F6 A          JMP     P1
213 FE81 0200 A          RTS     0
214 FE82          ;
215 FE82 0039 A G9:      .WORD   '9'/256
216 FE83 0007 A D7:      .WORD   7


217 FE84                 .PAGE
218 FE84          ;*************************************************************************
219 FE84          ;*                                                                       *
220 FE84          ;*       GETHX - GET VALUE OF 4 HEX CHARACTERS INTO AC1                  *
221 FE84          ;*                                                                       *
222 FE84          ;*       ON EXIT:  AC0=0  IF LAST CHARACTER WAS CR OR LF                 *
223 FE84          ;*                 AC0>0  IF LAST CHARACTER WAS COMMA OR COLON           *
224 FE84          ;*                 AC2 AND AC3 ARE UNDISTURBED                           *
225 FE84          ;*                                                                       *
226 FE84          ;*************************************************************************
227 FE84          ;
228 FF84 4D00 A GETHX:   LI      AC1,0           ; SUBROUTINE RETURNS HEX VALUE
229 FE85 29D1 A NC:      JSR     GECHO           ;    OF 4 TTY CHARS IN AC1
```

```
FE86 4C00 A          LI      AC0,0           ; RETURN AFTER SENDING CR,LF
FE87 E1FA A GETHXA:  SKG     AC0,G9          ; PROCESS CHARACTER
FE88 E114 A          SKG     AC0,GOM1
FE89 2104 A          JMP     CKAF
FE8A 6114 A EVAL:    AND     AC0,HF          ; CONVERT CHARACTER TO BINARY
FE8B 5004 A          SHL     AC1,4           ; SHIFT HIGHER DIGITS
FE8C 3182 A          RXOR    AC0,AC1         ; ADD NEW DIGIT TO NUMBER
FE8D 21F7 A          JMP     NC
FE8E E10D A CKAF:    SKG     AC0,GF          ; CHECK A TO F CHARACTERS
FE8F E10E A          SKG     AC0,GAM1
FE90 2102 A          JMP     CKT
FE91 4809 A          AISZ    AC0,9           ; CONVERT LOW 4 BITS TO A-F
FE92 21F7 A          JMP     EVAL
FE93 F106 A CKT:     SKNE    AC0,COMMA       ; NOT HEX DIGIT
FE94 0200 A          RTS     0               ; CHECK FOR TERMINATOR
FE95 F105 A          SKNE    AC0,COLON
FE96 0200 A          RTS     0
FE97 11FE A          BCC     ZRO,.-1
FE98 4400 A ABORT:   PULL    AC0             ; ABORT: POP STACK AND
FE99 218A A ERROR2:  JMP     ERROR           ; GO TO ERROR RETURN
FE9A                 ;
FE9A 002C A COMMA:   .WORD   ','/256
FE9B 003A A COLON:   .WORD   ':'/256
FE9C 0046 A GF:      .WORD   'F'/256
FE9D 002F A GOM1:    .WORD   '0'/256-1
FE9E 0040 A GAM1:    .WORD   'A'/256-1
FE9F 000F A HF:      .WORD   0F
FEA0 240E A JUMPH:   JMP     @RETADD         ; JUMP INSTRUCTION TO CONTINUE
FEA1 2186 A SCRLF:   JMP     CRLF            ; LONG JSR TO CRLF


FEA2                 .PAGE
FEA2                ;*************************************************************************
FEA2                ;*                                                                       *
FEA2                ;*       JSR     RANGE           GET 2 ADDRESSES FOR RANGE               *
FEA2                ;*                                                                       *
FEA2                ;*       ON EXIT:  AC3  = BEGINNING OF RANGE                             *
FEA2                ;*                 CWRD = END OF RANGE                                   *
FEA2                ;*                                                                       *
FEA2                ;*************************************************************************
FEA2                ;
FEA2 4600 A RANGE:   PULL    AC2             ; RETURN ADDRESS IS IN AC2
FEA3 29E0 A          JSR     GETHX           ; GET 1ST ARGUMENT
FEA4 A40A A          ST      AC1,CWRD        ; SAVE IT FOR NOW
FEA5 1101 A          BOC     ZRO,.+2         ; ONLY ONE ARGUMENT - MAKE BOTH THE SAME
FEA6 29DD A          JSR     GETHX
FEA7 8C0A A          LD      AC3,CWRD        ; AC3 HAS BEGINNING OF RANGE
FEA8 A40A A          ST      AC1,CWRD        ; CWRD HAS END OF RANGE
FEA9 3C81 A          RCPY    AC3,AC0         ; COMPARE ARGUMENTS
FEAA 5001 A          CAI     AC0,1
FEAB 3400 A          RADD    AC1,AC0         ; AC0 HAS END-BEGIN
FEAC E18F A          SKG     AC0,ZERO        ; IF END <= BEGIN THEN END := BEGIN
FEAD AC0A A          ST      AC3,CWRD
FEAE 2200 A          JMP     (AC2)           ; RETURN
```

```
                        .PAGE   'COMMAND PROCESSING'
                ;*******************************************************************
                ;*                                                                 *
                ;*      I DD,XX:YY                                                 *
                ;*                                                                 *
                ;*      MOVE DATA IN XX:YY UP ONE; THEN INSERT DD AT XX            *
                ;*                                                                 *
                ;*******************************************************************
                ;
FEAF 29D4 A INSERT: JSR     GETHX           ; GET FIRST ADDRESS
FEB0 11E8 A         BOC     ZRO,ERROR2
FEB1 A40D A         ST      AC1,DATA        ; SAVE DATA
FEB2 29EF A         JSR     RANGE           ; GET ADDRESS RANGE
FEB3 880D A         LD      AC2,DATA
FEB4 900A A LOOP:   LD      AC0,@CWRD       ; MOVE OLD DATA UP ONE
FEB5 780A A         ISZ     CWRD            ; INCREMENT ADDRESS FOR STORING
FEB6 B00A A         ST      AC0,@CWRD       ; STORE AT NEXT HIGHER ADDRESS
FEB7 7C0A A         DSZ     CWRD            ; RESTORE ADDRESS
FEB8 FC0A A         SKNE    AC3,CWRD        ; CHECK IF DONE YET
FEB9 2102 A         JMP     .+3
FEBA 7C0A A         DSZ     CWRD            ; DECREMENT ADDRESS POINTER
FEBB 21F8 A         JMP     LOOP
FEBC AB00 A         ST      AC2,(AC3)       ; INSERT THE DATA
FEBD 2190 A EXIT1:  JMP     EXIT
FEBE                ;
FEBE                ;
FEBE                ;*******************************************************************
FEBE                ;*                                                                 *
FEBE                ;*      N XXXXXX...             NOTE: INSERT COMMENTS              *
FEBE                ;*                                                                 *
FEBE                ;*******************************************************************
FEBE                ;
FEBE 2998 A NOTE:   JSR     GECHO           ; ECHO ALL CHARACTERS
FEBF 21FD A         JMP     EXIT1           ; (WITHOUT FURTHER PROCESSING)
FEC0 21FD A         JMP     NOTE


                        .PAGE
FEC1                ;*******************************************************************
FEC1                ;*                                                                 *
FEC1                ;*      TYPE MEMORY CONTENTS    XX THROUGH YY                      *
FEC1                ;*                                                                 *
FEC1                ;*      T XX:YY   OR   T XX                                        *
FEC1                ;*                                                                 *
FEC1                ;*******************************************************************
FEC1                ;
FEC1 29E0 A TYPE:   JSR     RANGE           ; GET ADDRESS RANGE
FEC2 29DE A LINE:   JSR     SCRLF           ; NEW LINE: CRLF FIRST
FEC3 3E81 A         RCPY    AC3,AC2
FEC4 29AF A         JSR     PUTW            ; TYPE ADDRESS
FEC5 4C20 A RTYP:   LI      AC0,' '/256
FEC6 29A7 A         JSR     SENDC           ; SEND TWO BLANKS
FEC7 8800 A SWRD:   LD      AC2,(AC3)       ; TYPE OUT VALUE
FEC8 29AB A         JSR     PUTW
FEC9 FC0A A         SKNE    AC3,CWRD        ; CHECK IF DONE YET
FECA 2106 A         JMP     FIN             ; FINISHED
FECB 1E05 A         BOC     JC14,FIN        ; TERMINATE IF ATTEMPTED TTY INPUT
FECC 4B01 A         AISZ    AC3,1           ; INCREMENT ADDRESS
FECD 3C81 A         RCPY    AC3,AC0         ; CHECK FOR END OF LINE
FECE 71B4 A         SKAZ    AC0,D7
FECF 21F7 A         JMP     SWRD
FED0 21F1 A         JMP     LINE
FED1                ;
FED1 29CF A FIN:    JSR     SCRLF           ; GIVE CR,LF WHEN FINISHED
FED2 21EA A         JMP     EXIT1           ; GO BACK TO PROMPT
FED3                ;
FED3                ;
```

```
                ;*********************************************************************
                ;*                                                                   *
                ;*      R                       TYPE OUT ALL REGISTERS               *
                ;*                                                                   *
                ;*      ORDER:   AC0  AC1  AC2  AC3  FLAGS  TOP-OF-STACK            *
                ;*                                                                   *
                ;*********************************************************************
                ;
2983 A REGTYP:  JSR     GECHO           ; WAIT FOR CR/LF
2100 A          JMP     .+1
4F02 A          LI      AC3,2           ; REGISTER VALUES STORED IN LOCS 2-5
4D07 A          LI      AC1,7           ; FLAGS AND STACK ARE LOC 6-7
A40A A          ST      AC1,CWRD
21EC A          JMP     RTYP


                .PAGE
                ;*********************************************************************
                ;*                                                                   *
                ;*      H XXXX                  SET BREAKPOINT ADDRESS AT XXXX       *
                ;*      H                       REMOVE BREAKPOINT                    *
                ;*                                                                   *
                ;*********************************************************************
                ;
900B A HALT:    LD      AC0,@HLOC       ; CHECK FOR PREVIOUS BREAKPOINT
840C A          LD      AC1,HDATA
B40B A          ST      AC1,@HLOC       ;    THENERESTORE ORIGINAL INSTRUCTION
29A6 A          JSR     GETHX           ; GET BREAKPOINT ADDRESS
3481 A          RCPY    AC1,AC0
A00B A          ST      AC0,HLOC        ; SAVE ADDRESS
11DC A          BCC     ZRO,EXIT1       ; ADDR=0 MEANS NO HALTS
940B A          LD      AC1,@HLOC       ; GET INSTRUCTION TO BE SAVED
A40C A          ST      AC1,HDATA
8589 A          LD      AC1,JUMPI       ; REPLACE WITH JUMP INSTRUCTION
B40B A          ST      AC1,@HLOC
21D7 A          JMP     EXIT1


                .PAGE
                ;*********************************************************************
                ;*                                                                   *
                ;*      GO: EXECUTE                                                  *
                ;*                                                                   *
                ;*      G XXXX                  BEGIN EXECUTION AT XXXX              *
                ;*      G                       BEGIN EXECUTION AT BREAKPOINT        *
                ;*                                                                   *
                ;*********************************************************************
                ;
299D A GO:      JSR     GETHX           ; GET RETURN ADDRESS
3481 A          RCPY    AC1,AC0         ; CHECK FOR ADDRESS=0
15C7 A          BOC     NZRO,JADDR
800B A          LD      AC0,HLOC        ; CONTINUE AFTER BREAKPOINT
11AE A          BOC     ZRO,ERROR2      ; ERROR IF NO PLACE TO GO
4801 A          AISZ    AC0,1
A00E A          ST      AC0,RETADD      ; STORE BREAKPOINT LOC + 1
81B2 A          LD      AC0,JUMPH
A00D A          ST      AC0,RETLOC      ; LOAD RETLOC WITH RETURN JUMP
4D0C A          LI      AC1,HDATA       ; EXECUTE THE SAVED INSTRUCTION
A40A A JADDR:   ST      AC1,CWRD        ; PUT JUMP ADDRESS IN CWRD
8403 A          LD      AC1,SAV1        ; RESTORE REGISTERS
8804 A          LD      AC2,SAV2
8C05 A          LD      AC3,SAV3
8007 A          LD      AC0,STACK       ; RESTORE TOP OF STACK
5400 A          XCHRS   AC0
7C08 A          DSZ     SELECT          ; RESTORE SELECT FLAG
2101 A          JMP     .+2
```

```
410 FEF8 0A00 A          SFLG    2
411 FEF9 8006 A          LD      AC0,FLAGS       ; RESTORE FLAGS
412 FEFA 4000 A          PUSH    AC0
413 FEFB 0280 A          PULLF
414 FEFC 8002 A          LD      AC0,SAV0
415 FEFD 240A A          JMP     @CWRD           ; EXECUTE
416 FEFE         ;
417 FFFE FE00 A          .END    BEGIN

 ******      0 ERRORS IN ASSEMBLY   ******


ABORT  AC0    AC1    AC2    AC3    ALTER  ALTER1 ALTER2 ALTER3 BEGIN
FE98 A 0000 A 0001 A 0002 A 0003 A FE3D A FE41 A FE44 A FE45 A FE00 A

BLNK   CKAF   CKCOM  CKT    COLON  COMMA  CPROC  CRLF   CTBL   CWRD
FE6C A FE8E A FE47 A FE93 A FE9B A FE9A A FE4C A FE28 A FE2C A 000A A

D7     DATA   ERROR  ERROR2 EVAL   EXIT   EXIT1  FIN    FLAGS  GOM1
FE83 A 000D A FE24 A FE99 A FE8A A FE4E A FEBD A FED1 A 0006 A FE9D A

G9     GALT   GAM1   GCR    GECHO  GETADD GETHX  GETHXA GF     GLF
FE82 A FE6B A FE9E A FE69 A FE57 A FE0E A FE84 A FE87 A FE9C A FE6A A

GO     GOTOIT H7F    HALT   HDATA  HF     HLOC   INSERT JADDR  JC14
FEE6 A FE21 A FE68 A FED9 A 000C A FE9F A 000B A FEAF A FEF0 A 000E A

JUMPH  JUMPI  LDREG  LINE   LOOP   NC     NEG    NOTE   NXCOM  NZRO
FEA0 A FE6D A FE4F A FEC2 A FEB4 A FE85 A 000B A FEBE A FE1A A 0005 A

P1     POINTE POS    PROMPT PUTW   RANGE  RECV   REGTYP RETADD RETLOC
FE77 A 0009 A 0002 A FE0D A FE74 A FEA2 A FE66 A FED3 A 000E A 000D A

RTYP   SAV0   SAV1   SAV2   SAV3   SCRLF  SELECT SEND   SENDC  STACK
FEC5 A 0002 A 0003 A 0004 A 0005 A FEA1 A 0008 A FE67 A FE6E A 0007 A

SWRD   TYPE   ZERO   ZRO
FEC7 A FEC1 A FE3C A 0001 A

   F5CA    9EC4
```